# Comment tester les logiciels

**- Projet d'étude et de recherche d'analyse -**

**Apprenti** : Jules PRINCE
**Maître d'apprentissage** : Sébastien BANON
**Tuteur Enseignant** : Florian Bridoux

**Entreprise** : Schneider Electric
09/2020 - 02/2024

9 février 2024

# Table des matières

# 1 Introduction

Dans le paysage informatique contemporain, la qualité des logiciels est un enjeu crucial, constituant le pilier fondamental sur lequel repose le fonctionnement efficace de nombreuses industries et activités humaines. En effet, les logiciels sont omniprésents dans notre quotidien, que ce soit à travers des applications mobiles qui facilitent nos interactions sociales, des systèmes de gestion d'entreprise qui optimisent nos processus organisationnels, ou encore des infrastructures informatiques critiques dans des domaines sensibles tels que la santé ou la finance. Dans ce contexte, la moindre défaillance peut avoir des conséquences désastreuses, allant de la perte de productivité à des situations mettant en jeu la sécurité des individus, voire la pérennité même des organisations.

Pour garantir la fiabilité, la robustesse et la performance des logiciels, il est impératif de les soumettre à un processus de test rigoureux. Les tests de logiciels sont bien plus qu'une simple formalité ou une étape à cocher dans le processus de développement ; ce sont des activités essentielles menées tout au long du cycle de vie d'un logiciel, depuis sa conception jusqu'à sa maintenance, visant à identifier les défauts, les erreurs et les comportements inattendus des applications informatiques.

Ce rapport a donc pour objectif d'effectuer une analyse approfondie sur ce sujet crucial. Dans un premier temps, nous explorerons l'importance vitale des tests logiciels et les problèmes qu'ils viennent résoudre, mettant en lumière leur rôle central dans la création de produits informatiques fiables et performants. Ensuite, nous plongerons dans l'examen de trois méthodologies de test essentielles : les tests unitaires, les tests d'intégration et les tests de validation et de vérification. En détaillant ces différentes approches, nous chercherons à mieux comprendre comment elles contribuent à l'assurance qualité des logiciels et à l'atteinte des objectifs de développement.

# 2 Importance des tests de logiciels

Nous allons voir dans cette partie les besoins supra importants de tests dans le contexte de développement logicielle.

## 2.1 Garantie de la fiabilité et de la robustesse du logiciel

L'assurance de la fiabilité et de la robustesse d'un logiciel est l'un des principaux objectifs des tests logiciels. En effet, dans un environnement où les logiciels sont de plus en plus complexes et interconnectés, la présence de défauts ou d'erreurs peut compromettre gravement le bon fonctionnement du système, entraînant des dysfonctionnements, des pertes de données ou même des situations mettant en danger la sécurité des utilisateurs.

Les tests de logiciels visent à identifier et à corriger les défauts dès les premières étapes du processus de développement, réduisant ainsi les risques de défaillance une fois le logiciel déployé en production. En détectant les bugs et les comportements inattendus, les tests permettent aux développeurs de résoudre les problèmes avant qu'ils ne deviennent des obstacles majeurs, garantissant ainsi une expérience utilisateur fluide et sans interruption.

La fiabilité d'un logiciel se traduit par sa capacité à fonctionner de manière prévisible et stable dans différentes conditions d'utilisation. Les tests de fiabilité évaluent la capacité du logiciel à maintenir ses performances et sa disponibilité dans des situations variées, telles que des charges de travail élevées ou des conditions environnementales changeantes. En identifiant les points faibles du système, les tests de fiabilité permettent aux développeurs d'améliorer la résilience du logiciel face aux éventuels incidents et de garantir sa disponibilité continue.

De même, la robustesse d'un logiciel se mesure à sa capacité à gérer les erreurs et les situations imprévues sans compromettre son fonctionnement global. Les tests de robustesse évaluent la capacité du logiciel à résister aux pannes, aux attaques malveillantes ou aux entrées incorrectes de l'utilisateur, en vérifiant sa capacité à se rétablir de manière autonome et à limiter les dommages potentiels.

En garantissant la fiabilité et la robustesse du logiciel, les tests contribuent à renforcer la confiance des utilisateurs dans le système, favorisant ainsi son adoption et sa pérennité sur le marché. De plus, en réduisant les risques de défaillance et en minimisant les coûts de maintenance, les tests de logiciels permettent aux organisations de maximiser leur retour sur investissement et de rester compétitives dans un environnement en constante évolution.

## 2.2 Identification précoce des défauts et des erreurs

L'un des principaux objectifs des tests de logiciels est d'identifier et de corriger les défauts et les erreurs dès les premières étapes du processus de développement. Cette approche, connue sous le nom de détection précoce des défauts, permet de réduire les coûts et les risques associés aux corrections ultérieures, tout en améliorant la qualité globale du produit final.

En détectant les défauts dès leur apparition, les tests de logiciels permettent aux développeurs de les traiter rapidement, avant qu'ils ne se propagent et n'engendrent des conséquences plus graves dans le système. Cela permet d'éviter l'accumulation de bogues et de réduire le temps nécessaire pour effectuer des corrections, ce qui contribue à maintenir le projet sur les rails et à respecter les délais de livraison.

De plus, l'identification précoce des défauts permet de minimiser les risques associés aux erreurs de conception ou aux décisions prises sur la base de données incorrectes. En identifiant les problèmes dès les premières

phases du développement, les tests de logiciels offrent aux équipes de développement la possibilité de réévaluer leurs choix et de prendre des mesures correctives avant que les erreurs ne deviennent coûteuses à corriger.

Enfin, l'identification précoce des défauts contribue à renforcer la confiance des parties prenantes dans le processus de développement et dans la qualité du produit final. En démontrant un engagement envers l'excellence et la rigueur dans les pratiques de test, les équipes de développement peuvent inspirer la confiance des clients, des utilisateurs finaux et des investisseurs, renforçant ainsi la réputation de l'organisation et favorisant sa réussite sur le marché.

En conclusion, l'identification précoce des défauts et des erreurs est un aspect essentiel des tests de logiciels, offrant des avantages significatifs en termes de réduction des coûts, de gestion des risques et d'amélioration de la qualité du produit final. En investissant dans des activités de test rigoureuses dès les premières phases du développement, les organisations peuvent créer des logiciels fiables, performants et conformes aux attentes des utilisateurs, assurant ainsi leur succès à long terme.

## 2.3 Réduction des risques et des coûts de maintenance

Les tests de logiciels jouent un rôle crucial dans la réduction des risques et des coûts associés à la maintenance des systèmes informatiques. En identifiant et en corrigeant les défauts dès les premières phases du développement, les tests contribuent à minimiser les risques de défaillance du logiciel une fois déployé en production.

La détection précoce des défauts permet de réduire les risques opérationnels en anticipant et en prévenant les situations potentiellement dommageables pour le système et ses utilisateurs. En identifiant les failles de sécurité, les bugs fonctionnels et les erreurs de conception dès leur apparition, les tests de logiciels permettent aux équipes de développement de prendre des mesures correctives avant que ces problèmes ne se transforment en incidents coûteux ou en crises majeures.

De plus, en réduisant le nombre de défauts et d'erreurs dans le logiciel, les tests contribuent à diminuer les coûts de maintenance associés à leur correction. Les corrections de bugs après le déploiement sont souvent beaucoup plus coûteuses et complexes que celles effectuées pendant les phases de développement, car elles nécessitent souvent des efforts de débogage plus importants, des ressources supplémentaires et des interruptions potentielles des opérations en cours.

En investissant dans des activités de test rigoureuses tout au long du cycle de développement, les organisations peuvent donc réaliser des économies significatives en termes de coûts de maintenance. Les tests permettent d'identifier les problèmes dès leur apparition, ce qui réduit les besoins en ressources de correction ultérieure et minimise les perturbations pour les utilisateurs finaux.

En conclusion, la réduction des risques et des coûts de maintenance est l'un des principaux bénéfices des tests de logiciels. En identifiant et en corrigeant les défauts dès les premières phases du développement, les tests permettent aux organisations de minimiser les risques opérationnels, d'optimiser l'efficacité des processus de maintenance et de garantir la fiabilité et la stabilité des systèmes informatiques sur le long terme.

# 3 Méthodologies de test

Pour répondre aux problèmes énoncé précédemment nous allons voir plusieurs méthodologie de tests pour améliorer la qualités des logiciels.

## 3.1 Tests unitaires

### 3.1.1 Définition et objectifs

Les tests unitaires représentent une méthode de test logiciel qui cible la vérification individuelle du bon fonctionnement de chaque composant ou unité de code d'un logiciel. Ces unités de code peuvent être des fonctions, des méthodes, voire même des classes dans le cas de la programmation orientée objet. L'objectif premier des tests unitaires réside dans la confirmation que chaque unité opère comme prévu, générant les résultats attendus pour différentes entrées.

Typiquement, les développeurs rédigent les tests unitaires, souvent avant la finalisation de l'implémentation du code. Ces tests sont automatisés, ce qui permet de les exécuter de manière régulière et rapide à chaque modification du code. Cette automatisation facilite l'identification des erreurs ou des changements de comportement indésirables dès leur apparition, offrant ainsi la possibilité de les corriger avant qu'ils ne prennent de l'ampleur dans le système.

Les objectifs majeurs des tests unitaires comprennent la validation de la logique du code. En vérifiant chaque unité de code, ils s'assurent que celles-ci fonctionnent conformément à la logique définie par le développeur, englobant ainsi la vérification des divers chemins d'exécution possibles à l'intérieur de chaque unité.

De plus, les tests unitaires visent la détection précoce des erreurs. En identifiant les problèmes au niveau des unités individuelles, ils permettent de repérer les erreurs dès leur apparition, facilitant ainsi leur correction rapide et minimisant les risques de propagation dans d'autres parties du système.

Enfin, les tests unitaires contribuent au maintien de la qualité du code. En encourageant les bonnes pratiques de programmation telles que la modularité, la réutilisabilité et la séparation des préoccupations, ils incitent les développeurs à produire un code de meilleure qualité et plus robuste.

### 3.1.2 Processus de mise en œuvre

Le processus de mise en œuvre des tests unitaires implique plusieurs étapes clés pour garantir leur efficacité et leur intégration réussie dans le cycle de développement logiciel.

- **Identification des unités à tester :** La première étape consiste à identifier les différentes unités de code à tester. Il s'agit généralement des fonctions, des méthodes ou des classes qui représentent les blocs de base de la logique de l'application. Cette étape nécessite une compréhension approfondie de la structure du code et de ses interactions.
- **Écriture des tests unitaires :** Une fois les unités identifiées, les développeurs écrivent des cas de test pour chaque unité. Les cas de test doivent couvrir différents scénarios d'utilisation et inclure une variété d'entrées et de conditions de test. Les frameworks de tests unitaires, tels que JUnit pour Java ou NUnit pour .NET, sont souvent utilisés pour faciliter l'écriture et l'exécution des tests.
- **Exécution des tests :** Les tests unitaires sont exécutés régulièrement, idéalement à chaque modification du code ou lors de la compilation du projet. Cette automatisation garantit que les tests sont exécutés de manière cohérente et rapide, ce qui permet une détection précoce des erreurs.

- **Analyse des résultats :** Une fois les tests exécutés, les résultats sont analysés pour identifier les éventuels échecs ou comportements inattendus. Les outils de gestion de tests, tels que Jenkins ou Travis CI, peuvent être utilisés pour suivre les résultats des tests et notifier les développeurs en cas d'échec.
- **Correction des erreurs :** En cas d'échec des tests, les développeurs corrigent les erreurs identifiées et réécrivent les tests si nécessaire. Il est important de s'assurer que les tests échouent en raison d'un comportement incorrect du code et non en raison d'un défaut dans le test lui-même.
- **Intégration continue :** Les tests unitaires font partie intégrante du processus d'intégration continue, où les modifications de code sont intégrées et testées automatiquement de manière continue. Cela garantit que le code reste fonctionnel et fiable à chaque étape du développement.

En suivant ce processus de mise en œuvre, les tests unitaires peuvent contribuer de manière significative à la qualité et à la stabilité du code, en permettant une détection précoce des erreurs et en encourageant les bonnes pratiques de développement logiciel.

### 3.1.3 Avantages et limitations

Les tests unitaires sont un élément essentiel du processus de développement logiciel, offrant à la fois des avantages significatifs et présentant des limitations à prendre en compte.

D'un côté, les tests unitaires offrent l'avantage indéniable de détecter précocement les erreurs. En vérifiant le bon fonctionnement des unités de code à un niveau granulaire, ils permettent d'identifier les défauts dès les premières phases du développement, facilitant ainsi leur correction rapide avant qu'ils ne deviennent des problèmes plus complexes et coûteux à résoudre. Cette approche contribue également à réduire les risques de régression en s'assurant que les nouvelles fonctionnalités ou corrections n'ont pas d'effets indésirables sur le reste du système. De plus, en fournissant une documentation vivante du comportement attendu de chaque unité de code, les tests unitaires facilitent la compréhension du code par les développeurs et contribuent à améliorer sa qualité globale en encourageant les bonnes pratiques de développement.

Cependant, les tests unitaires présentent également des limitations qu'il est important de prendre en compte. Tout d'abord, ils offrent une couverture limitée des interactions entre les unités de code, ce qui signifie qu'ils peuvent ne pas détecter certains types d'erreurs ou de comportements indésirables qui se produisent uniquement lorsque plusieurs unités interagissent. De plus, écrire et maintenir des tests unitaires nécessite un investissement de temps et de ressources supplémentaire de la part des développeurs, ce qui peut être perçu comme un coût initial plus élevé, surtout dans les phases initiales du projet. De plus, comme le code évolue, les tests unitaires doivent également être mis à jour pour refléter les changements dans le code, ce qui peut nécessiter des efforts de maintenance supplémentaires pour garantir leur efficacité au fil du temps. Enfin, bien qu'avoir une suite de tests unitaires complète puisse offrir un certain niveau de sécurité, elle ne garantit pas l'absence totale de défauts ou d'erreurs dans le système, ce qui nécessite des méthodes de test complémentaires pour assurer une couverture complète et une qualité globale du produit logiciel.

En conclusion, bien que les tests unitaires offrent de nombreux avantages pour garantir la qualité et la stabilité du code, il est important de reconnaître leurs limitations et de les utiliser de manière judicieuse en complément d'autres méthodes de test pour assurer une couverture complète et une qualité globale du produit logiciel.

## 3.2 Tests d'intégration

### 3.2.1 Objectifs et principes

Les tests d'intégration représentent une étape majeure dans le processus de développement logiciel, s'attachant à garantir le bon fonctionnement des différentes parties d'un système une fois qu'elles sont combinées. Contrairement aux tests unitaires qui se concentrent sur la vérification du fonctionnement des unités de code individuelles, les tests d'intégration évaluent la cohérence et la compatibilité des interactions entre ces unités, formant ainsi un système complet et fonctionnel.

L'objectif principal de ces tests est de vérifier l'interopérabilité des composants. Ils s'assurent que les différents modules ou composants du système communiquent correctement, en respectant les spécifications et interfaces définies. Par cette validation, ils garantissent que les composants interagissent harmonieusement, évitant les conflits potentiels qui pourraient compromettre le bon fonctionnement du système.

Un autre objectif majeur des tests d'intégration est d'identifier et de résoudre les erreurs de communication. En simulant les interactions entre les divers composants, ces tests permettent de déceler les problèmes tels que les délais de réponse excessifs, les pertes de données ou les incompatibilités de protocole. Ainsi, ils assurent une circulation fluide des données et des instructions entre les différentes parties du système.

En adoptant une approche progressive, les tests d'intégration sont réalisés par étapes successives. Les différentes parties du système sont intégrées progressivement, permettant de détecter et de corriger les problèmes d'interopérabilité dès leur apparition. Cette méthode, en plus de réduire les risques de régression, contribue à renforcer la fiabilité et la robustesse du système dans son ensemble.

En somme, les tests d'intégration jouent un rôle crucial dans le processus de développement logiciel, en assurant la cohérence et la compatibilité des interactions entre les composants du système, et en garantissant la qualité et la fiabilité du produit final. Par leur approche progressive et leur focus sur l'interopérabilité, ils constituent une pièce maîtresse pour la réussite d'un projet logiciel.

### 3.2.2 Approches de test

Les tests d'intégration impliquent l'utilisation de diverses approches pour garantir le bon fonctionnement du système dans son ensemble. Parmi ces approches, on retrouve :

- **Intégration ascendante :** Cette méthode commence par intégrer les composants de bas niveau, puis progresse vers les niveaux supérieurs. Cela permet de tester les parties fondamentales du système avant d'ajouter des fonctionnalités plus complexes.
- **Intégration descendante :** Contrairement à l'approche ascendante, cette méthode débute par l'intégration des composants de haut niveau, puis descend progressivement vers les niveaux inférieurs. Elle permet de tester les interactions entre les composants principaux avant d'ajouter des détails plus spécifiques.
- **Intégration par modules :** Cette approche consiste à regrouper les composants similaires en modules et à les tester ensemble. Cela permet de vérifier le bon fonctionnement des fonctionnalités spécifiques du système.
- **Tests de bout en bout :** Ces tests évaluent le système dans son ensemble, en simulant le flux complet des données et des actions depuis le début jusqu'à la fin. Ils permettent de vérifier la cohérence et l'intégration de toutes les parties du système.

Chaque approche de test d'intégration présente des avantages et des inconvénients, et le choix de la méthode appropriée dépend des besoins spécifiques du projet et de la structure du système à tester.

### 3.2.3 Gestion des dépendances et des interactions

La gestion des dépendances et des interactions est un aspect crucial des tests d'intégration. Cette phase vise à garantir que toutes les interactions entre les composants du système sont correctement gérées et que les dépendances entre ces composants sont clairement définies.

L'une des principales tâches de la gestion des dépendances et des interactions est de s'assurer que les composants sont intégrés dans le bon ordre. Cela signifie que les composants qui dépendent d'autres doivent être intégrés après leurs dépendances, afin de garantir un fonctionnement correct du système.

De plus, cette phase implique souvent la simulation ou la mise en place d'environnements de test spécifiques pour reproduire les interactions réelles entre les composants. Cela peut inclure l'utilisation de simulateurs, de stubs ou de mocks pour remplacer les composants externes ou les services tiers, permettant ainsi de contrôler l'environnement de test et de s'assurer que les interactions sont correctement gérées.

Enfin, la gestion des dépendances et des interactions comprend également la vérification des effets de bord et des interactions imprévues entre les composants. Il s'agit de s'assurer que les modifications apportées à un composant n'affectent pas de manière inattendue d'autres parties du système, ce qui pourrait entraîner des erreurs ou des comportements indésirables.

Dans l'ensemble, la gestion des dépendances et des interactions est une étape essentielle des tests d'intégration, permettant de garantir que le système fonctionne de manière cohérente et fiable en tenant compte de toutes les interactions entre ses composants.

## 3.3 Tests de validation et de vérification

### 3.3.1 Évaluation de la conformité aux exigences

Les tests de validation et de vérification sont des processus essentiels pour s'assurer que le logiciel développé répond aux exigences spécifiées. L'évaluation de la conformité aux exigences consiste à vérifier si le logiciel répond aux attentes définies dans les spécifications fonctionnelles et non fonctionnelles.

Dans cette phase, les tests sont conçus pour évaluer chaque fonctionnalité du logiciel par rapport aux exigences fonctionnelles établies. Il s'agit de s'assurer que le logiciel effectue les tâches spécifiées de manière précise et efficace, en tenant compte des différents scénarios d'utilisation prévus.

Par exemple, si une exigence fonctionnelle spécifie que le logiciel doit être capable de calculer des montants de facturation avec une précision de deux décimales, les tests de validation et de vérification vérifieront que cette fonctionnalité est correctement implémentée et que les résultats produits sont conformes à cette exigence.

De plus, cette phase de test examine également la conformité aux exigences non fonctionnelles telles que la performance, la sécurité, la convivialité et la compatibilité. Par exemple, les tests de performance évalueront la capacité du logiciel à répondre aux exigences de charge et de temps de réponse spécifiées, tandis que les tests de sécurité vérifieront que le logiciel protège les données sensibles conformément aux exigences de sécurité définies.

En résumé, l'évaluation de la conformité aux exigences dans les tests de validation et de vérification est essentielle pour garantir que le logiciel répond aux attentes définies dans les spécifications. Cela permet de s'assurer que le produit final est fonctionnel, fiable, sécurisé et conforme aux besoins et aux attentes des utilisateurs.

### 3.3.2 Simulation des scénarios d'utilisation réels

Une composante essentielle des tests de validation et de vérification consiste à simuler des scénarios d'utilisation réels pour évaluer le comportement du logiciel dans des conditions proches de celles rencontrées par les utilisateurs finaux.

Cette approche permet de mettre le logiciel à l'épreuve dans des situations représentatives de son utilisation quotidienne, ce qui permet de détecter les éventuels problèmes qui pourraient survenir dans des conditions réelles.

Dans le cadre du développement du firmware pour nos automates programmables, les tests de validation se concentreront sur le comportement d'une application cliente sur le système. Par exemple, si l'application du client concerne le contrôle des vannes d'eau, il s'agira de vérifier que l'application réagit correctement en fonction des valeurs des capteurs en activant les différentes vannes.

La simulation de scénarios d'utilisation réels permet également d'identifier les points de friction potentiels pour les utilisateurs, tels que des temps de chargement trop longs, des fonctionnalités difficiles à utiliser ou des erreurs de manipulation. Cela permet aux développeurs d'apporter des ajustements pour améliorer l'expérience utilisateur et garantir la satisfaction des utilisateurs finaux.

En résumé, la simulation des scénarios d'utilisation réels est une étape cruciale des tests de validation et de vérification, permettant d'évaluer le comportement du logiciel dans des conditions proches de celles rencontrées par les utilisateurs finaux. Cela contribue à garantir que le logiciel répond aux attentes des utilisateurs et fonctionne de manière fiable et efficace dans toutes les situations d'utilisation prévues.

### 3.3.3 Importance de l'expérience utilisateur dans les tests

L'expérience utilisateur (UX) joue un rôle crucial dans les tests de validation et de vérification des applications, notamment dans le cas des applications destinées à être utilisées par des clients sur des automates programmables.

Une interface utilisateur intuitive et conviviale est essentielle pour garantir que les utilisateurs peuvent interagir efficacement avec l'application et accomplir leurs tâches de manière fluide. Dans le cadre des tests, il est important d'évaluer non seulement la fonctionnalité de l'application, mais aussi son ergonomie et sa facilité d'utilisation.

Les tests doivent simuler les scénarios d'utilisation réels et évaluer la réaction des utilisateurs face à l'interface, en tenant compte de facteurs tels que la clarté des instructions, l'organisation des éléments à l'écran et la facilité de navigation. Les retours des utilisateurs doivent être pris en compte pour identifier les éventuels points de friction ou de confusion et apporter les ajustements nécessaires pour améliorer l'expérience utilisateur.

En fin de compte, une expérience utilisateur positive est un facteur déterminant dans l'adoption et l'utilisation continue d'une application. Les tests de validation et de vérification doivent donc accorder une attention particulière à cet aspect pour garantir la satisfaction et la fidélisation des clients.

# 4 Conclusion

En conclusion, les tests de logiciels sont une étape cruciale dans le processus de développement, visant à garantir la qualité, la fiabilité et la performance des applications informatiques. Dans ce rapport, nous avons examiné l'importance des tests de logiciels, mettant en lumière leur rôle dans la détection précoce des défauts, la réduction des risques et des coûts de maintenance, ainsi que dans la garantie de la fiabilité et de la robustesse du logiciel.

Nous avons également exploré trois méthodologies de test essentielles : les tests unitaires, les tests d'intégration et les tests de validation et de vérification. Chacune de ces méthodologies offre des avantages uniques et complémentaires, contribuant à assurer la qualité et la fonctionnalité des logiciels.

En outre, nous avons souligné l'importance de prendre en compte l'expérience utilisateur dans les tests, en reconnaissant son rôle central dans la satisfaction des clients et l'adoption réussie des applications.

En somme, les tests de logiciels sont un investissement essentiel pour les entreprises, leur permettant de livrer des produits de haute qualité, de répondre aux besoins des utilisateurs et de rester compétitives sur le marché. En continuant à accorder une attention particulière aux tests de logiciels et en adoptant les meilleures pratiques en la matière, les organisations peuvent garantir le succès de leurs projets informatiques et la satisfaction de leurs clients.